View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

BOOK REVIEW: *Prolog for Programmers*, by Feliks Kluzniak and Stanislaw Szpakowicz*

Prolog for Programmers is intended as an introduction to Prolog for people familiar with conventional programming languages. Reflecting this, in Chapter 1, Prolog is explained in terms of programming constructs and techniques found in Pascal. However, explaining Prolog in terms of Pascal sometimes becomes contrived, for example, backtracking is explained in terms of error recovery and unification is explained by giving a Pascal unification algorithm. Unfortunately, both the logical and extralogical features of Prolog are introduced together, blurring the distinction between the declarative and nondeclarative aspects of the language.

The second chapter is an introduction to mathematical logic and resolution intended to provide the reader with a theoretical understanding of Prolog. However, there is no mention of important concepts such as negation-as-failure. This chapter does contain a good discussion about search spaces and intelligent backtracking, though the statement

> The fact that Prolog can be used as a practical language is still largely due to our dexterity in fighting exponential complexity with the cut.

might be disputed by some.

The third chapter provides a useful and lucid introduction to a metamorphosis grammars and their practical use through the development of both a parser and code generator for small programming language.

Some useful programming techniques are given in the fourth chapter that includes programs to manipulate trees, lists, open lists, and d-lists. It also contains valuable techniques to reduce program memory requirements which were previously found only in Prolog's folklore.

The book also contains two large programming examples: David Warren's Warplan; and a toy-SEQUEL interpreter for a relational database. Janusz S. Bien has contributed a section briefly describing the Prolog dialects, Prolog I, Prolog II, M-Prolog, and micro-Prolog.

*Academic, 1985, 400 pp.

The remainder of the book (160 pages) is devoted to details of Prolog's implementation and includes the complete Pascal code for an interpreter for the authors' Prolog system, Toy. One questions the value of this for the intended audience.

Although the book does contain many useful techniques, there is no mention of the Prolog interpreter for Prolog; surely important for the serious Prolog programmer. Although the authors acknowledge that Prolog is increasingly being used to develop expert systems, there is no discussion of expert systems and their implementation in Prolog. A case study developing an Apes style expert system shell would seem both relevant and useful for the book's audience.

In summary, this book contains some quite useful practical information for both the novice and experienced Prolog programmer. However, we believe too much attention is paid to clarifying Prolog's syntactic details rather than its underlying concepts. This hides the real concepts underlying Prolog which any newcomer to Prolog, experienced programmer or not, needs to know.

Richard Helm
Kim Marriott
Dept. of Computer Science
University of Melbourne